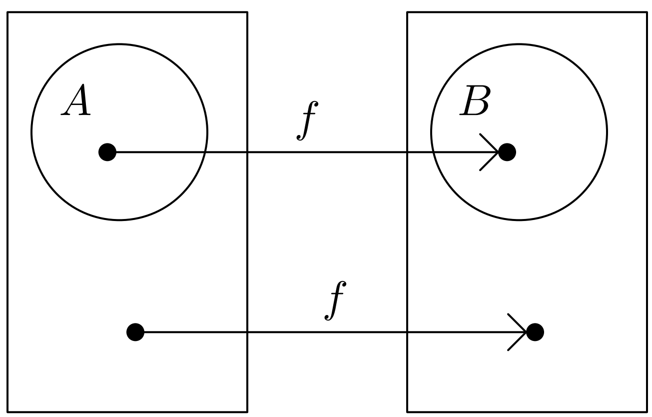
Reductions

We prove questions are undecidable by showing that answering the new question would enable one to decide a question we already know is undecidable.

Reductions

Recall that a T-computable function is a function from strings to strings for which there is a TM. Let A, B be languages. We say that A is ***reducible*** to B, written $A \leq_m B$, if there is a T-computable function f such that $w \in A$ exactly when $f(w) \in B$.



Reductions Preserve Hardness

Fact.
a) If A is reducible to B and B is recursive, then A is recursive.
b) If A is reducible to B and A is not recursive, then B is not recursive.

Proof (of a). Let TM R decide language B, and let function f reduce A to B. Construct TM S as follows: On input w, it computes $f(w)$ and submits this to R; then it accepts if R accepts. So S decides A.

Why The Notation $\leq$?

The above fact shows if one writes $A \leq_m B$, then B is as least as hard as A. This relationship behaves as one would expect. For example:

Fact. *For any languages A, B and C: If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.*

If f reduces A to B and g reduces B to C, then h defined by $h(w) = g(f(w))$ reduces A to C.

Practice

Show that for any languages A and B: If $A \leq_m B$ then $\bar{A} \leq_m \bar{B}$.

Solution to Practice

The same reduction works! If function f reduces A to B, then it maps A to B and $\bar{A}$ to $\bar{B}$.

State-Use is Undecidable

Consider the problem of determining whether a TM on input w ever enters a particular state q (called the ***state-use*** problem).

We reduce the acceptance problem A_{tm} to this.

State-Use is Undecidable

Suppose one has algorithm for state-use problem. Then modify it into an algorithm for A_{tm}: Take input $\langle M, w\rangle$ to the acceptance problem. Then introduce a new state q' and adjust M so that any transition leading to h_a leads to q' instead. Then answering whether M uses q' on w is equivalent to answering whether M accepts w. This we know is undecidable.

Acceptance of Blank Tape is Undecidable

$A_{bt} = \{ \langle M \rangle : M$ **accepts** $\varepsilon \}$ **is not recursive.**

The proof is to reduce A_{tm} **to** A_{bt}**.**

Acceptance of Blank Tape is Undecidable

The proof is to reduce A_{tm} to A_{bt}. That is, given TM M and string w, we build new TM M_w. The reduction f is $f(\langle M, w\rangle) = \langle M_w\rangle$ where M_w is programmed to:

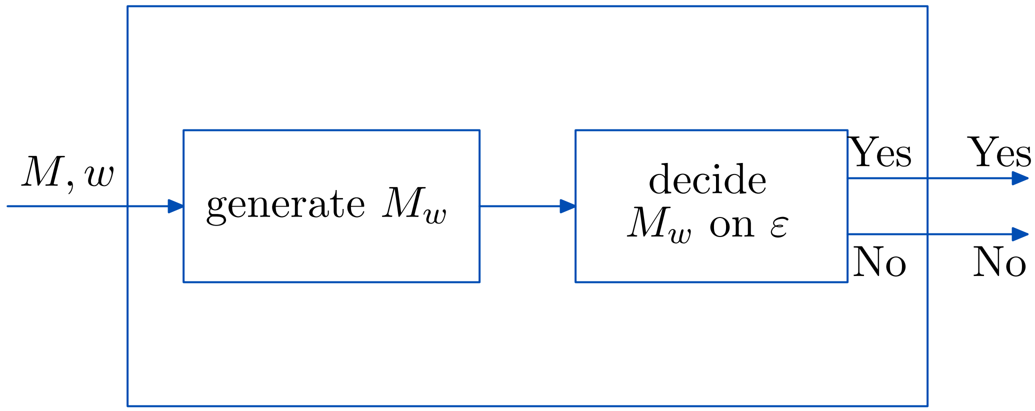
> (1) erase its input; (2) write w on the tape; (3) pass it to M; and (4) accept exactly when M accepts.

So M_w accepts ε exactly when $\langle M, w\rangle \in A_{tm}$.

Conclusion

Hence, if we could answer questions about A_{bt}, we would be able to answer questions about A_{tm}, which we know is undecidable.

Here is a visualization: the outer box does A_{tm} if we have a decider for A_{bt}.



Practice

Show that it is undecidable whether a TM ever writes a particular symbol on the tape.

Solution to Practice

Assume we have TM M and string w. Construct a new machine M_w. The TM M_w is programmed to erase its input, write w on the tape, and pass this over to M. If M accepts, then M_w writes a special symbol, say $\$$ on the tape. Thus if one could answer the question whether M_w writes $\$$ or not, one would be able to decide A_{tm}, which is undecidable.

Rice's Theorem

Actually, most questions about TMs are undecidable:

Rice's Theorem. *Any question about r.e. languages that is nontrivial is undecidable.*

Nontrivial means there is some language for which the answer is "yes" and some for which the answer is "no". We omit the beautiful but simple reduction.

Summary

A reduction is a mapping that preserves membership. A reduction can be used to show that one problem is undecidable given the undecidability of another problem. Some problems about TMs are proven undecidable by reduction from the acceptance problem A_{tm}.